https://doi.org/10.48009/2_iis_2006_293-297

METAPHORS GONE WILD: THE ILLUSIVE MACHINE CYCLE

Donald A. Carpenter, Mesa State College, dcarpent@mesastate.edu
Donna McAlister Kizzier, Morehead State University, d.kizzier@moreheadstate.edu

ABSTRACT

Information technology (IT) professionals use metaphors extensively to reduce complexity for the novice user. Some metaphors are good; others, not. Some are clear; others, confusing. This paper explores several metaphors, then examines in depth one particular metaphor that may be harmful in its present inaccurate portrayal, that of the "machine cycle." The goal is to enhance understanding of that metaphor and to encourage discretion in its use.

Keywords: Computer Literacy, Machine Cycle, Fetch-Execute Cycle, Metaphors

INTRODUCTION

Metaphors can be a springboard for comprehension as novices dive into computer literacy. Wide ranges and varieties of metaphors are used within computer literacy courses to enhance students' understanding. Indeed, a computer itself is a metaphor for a human. Efforts to develop automatic computation, dating to the seventeenth century, shared the common goal to "replace the facilities of human intellect" [15]. While the earliest devices up through the first true computers in the middle of the twentieth century focused on tabulation, calculation, and storage of data, the intention of the first computer scientists, like Alan Turing, remained the same: "We may hope that machines will eventually compete with men in all purely intellectual fields" [10]. While scientists in computing disciplines trudge steadily toward lofty goals of artificial intelligence, a quite profitable IT industry has matured by offering hardware and software that, over time, are closer and closer realizations of that original metaphor.

As the general metaphor of a computer as a human becomes closer to reality, the IT field uses many other figures of speech to explain aspects of the computer and its use. Some of the metaphors are sharp and quite effective. Some are good as far as they go, but incomplete. Others are marginal but widely used. Some metaphors are weak, but endure even to the point of becoming idioms. Some are premature, but allowed room for reality to grow into the analogy. Other metaphors are misunderstood and misused, but benign. In the following section, this paper presents examples to support the above contentions.

Subsequent to that, the paper turns its attention to a more serious matter: the downside of metaphors. Are there metaphors that are harmful in that they inhibit rather than foster understanding? Can a metaphor become out of control? This paper intends to show how one metaphor—in particular, that which is often described as the "machine cycle"—unfortunately answers both those questions in the affirmative.

THE GOOD, THE BAD, AND THE CONFOUNDED

The notion of an operating system providing an icon of a "trash can" or "recycle bin" is an example of a rather good metaphor that promotes comprehension. It is simple for a new computer operator to understand: move an unwanted file to a trash can and it is disposed. The metaphor is incomplete only in a desirable sense: if a disposed file is retrieved from the trash can, at least it does not have to be cleaned of yesterday's commingled coffee grounds.

Another effective metaphor that predates the personal computer era is the concept of documents, files and folders. The addition of clickable icons to represent those elements has enriched the metaphor. If one was to look for fault in that comparison, it might lie in that the metaphor is not extended to an effective comparison with reality. Rather than folders being components of hanging files, and, in turn, file drawers, file cabinets and a file room, computer folders can be placed in folders, which can be placed in folders, which can be placed in folders, et cetera. While the metaphor is incomplete, it is effective nevertheless.

Conversely, it is difficult to argue for the goodness of the computer "mouse" as a metaphor or even as a simile. The computer mouse is not functionally similar in any regard to its live namesake. Even the physical similarities are minimal. Yet, the term has caught on, perhaps because it is cute or because it provided some degree of comfort to a new computer user. Or perhaps it was simply a case of users not being offered an alternative term for that popular pointing device.

Similarly, the term "desktop" is, at best, a weak metaphor as a descriptor of the backdrop to the graphical user interface of an operating system. This author cannot think of the top of any desk where the worker has placed wallpaper or arranged icons to click, or buttons to push, in order to begin tasks. A better term might have been "control panel," but that was used for another purpose. Nonetheless, "desktop" endures to the point of having become an idiom, with instructors commonly using phrases such as "return to your desktop."

One metaphor that was premature was that of the "information superhighway." Of course, the initial uses of the term were in a predictive, prescriptive or wishful sense [3]. Yet, the term took hold even though the reality that it espoused was a far cry from the metaphor's description [2]. At that time, the Internet did not have easy on/off-ramps, universally understood regulations and signage, or a fast lane for the experienced user. Subsequently, however, the Internet has lived more fully into its information superhighway metaphor, complete with frequent construction zones and closures. However, it is much simpler just to use the term "the Internet," so "information superhighway" has become less often used, and thereby, ineffective.

The World Wide Web is a metaphor that might be understood differently by the general public than by IT professionals. The latter appreciates the description of a huge interconnection of documents encoded in hypertext mark up languages that are accessed by hypertext transfer protocols using an interconnection of physical and logical networks. By contrast, much of the public simply seems to think that the World Wide Web is another name for the Internet. As a universally understood metaphor, "World Wide Web" misses the mark.

So, not all metaphors are good ones. Some are weak, but benignly so. Other metaphors are simply confounded by their immaturity or by public misunderstanding or misuse.

THE MACHINE CYCLE

Are there metaphors that can be harmful in that they inhibit comprehension? Can a metaphor become out of control? This section addresses those questions by examining the case of the "machine cycle."

The Basics

John von Neumann is credited as the first to record results of research and conversations with others, especially those working on the ENIAC computer at University of Pennsylvania's Moore School of Electrical Engineering, regarding a "stored program concept" [15] which led to development of the general purpose computer. Using an analogy to the human brain, in 1945 von Neumann described a "*central arithmetic* part (CA)" [which today has evolved to be known as the arithmetic logic unit (ALU)], a "*central control* part (CC)" [today, the control unit], a super part "C" which is CA and CC together [today, central processing unit (CPU)], a "*memory* (M)" [today, main memory], "an outside recording medium (R)" [today, storage, input and output devices], and a "fixed clock, which provides certain stimuli that are necessary for its functioning at definite periodically recurrent moments" [11].

Wilkes clarified and expanded upon von Neumann's concepts with his discussion of registers and the complexity of the control unit [14]. The procedures by which CA, CC, and M interact in the von Neumann architecture have become known generally within computer science as "fetch-execute cycle" [1, 5], which is triggered by periodic firings of the clock. The fetch-execute cycle is also known as "machine cycle" [6], "instruction cycle" [12], and sometimes as "fetch-decode-execute cycle" [13].

The following statements summarize the commonality among the various descriptions by those authorities. The control unit fetches a machine language instruction from main memory and then passes the instruction to be executed by the ALU. Those who include "decode" as a separate step point out the control unit needs to figure out exactly what the instruction is that needs to be executed, e.g., an addition or a subtraction.

In actuality, the terms "fetch" and "execute" are each metaphors that are intended to add understanding for the novice. Conversely, the term "decode" extends the metaphor but perhaps confounds understanding, since in computer science circles, decode is part of the fetching phase.

Fetch is an active verb and implies something is brought back, just as Jack and Jill had to go up a hill to fetch a pail of water. The implication is that the control unit "goes to" memory and "brings back" a program instruction to a register. Yet, the control unit doesn't move and can't bring back. Rather the control unit makes an electronic copy of an instruction into a specialized unit of memory, known as the instruction register. The instruction of which it makes a copy is indicated by a pointer contained in the program register. "Fetch" certainly helps a novice to

understand more easily than does such a detailed explanation.

Likewise, the arithmetic logic unit doesn't execute an instruction in a human sense. Rather electricity flows through the appropriate circuits according to the pattern dictated by the electronic pulses of the fetched instruction. It should be noted that the fetched instruction is not always associated with the circuitry of the ALU, which handles only arithmetic and logical instructions. Storage, input and output instructions are handled by the controlling circuitry for the appropriate storage, input or output device. Of course, "execute" makes more sense to a novice than does a more complete and precise description, such as stated above.

Furthermore, the control unit does not necessarily need to decode an instruction as a separate step. The pattern of electronic pulses of the instruction itself serves as a trigger to the circuitry that will handle the instruction. However, a novice without an understanding of electronics could be baffled by a detailed explanation. Hence "decode" might itself be a good metaphor.

In actual practice, each of the two stages of fetch and execute can have multiple components depending on the nature of the particular central processing unit and the wording choice of the authority. For instance, the program register must be incremented so that the next logical instruction gets fetched. Another example is when there is an indirect address associated with an instruction, in which case that indirect address must be understood before the execution can begin. Both of those are extensions of the fetching process.

Moreover, a particular machine language instruction might require intermediate or final results to be placed in dedicated registers. Similarly, a particular architecture such as reduced instruction set computing (RISC) might specify additional sub-steps such as writing results to a buffer. Furthermore, in a parallel processing environment, execution of a particular instruction might be turned over to a separate processor, really complicating the explanation of the process. Those all are extensions of the execution process.

It would be a faux pas to not mention that all the discussions thus far pertain to general purpose digital computers. Analog computers and special purpose digital computers might use different processes altogether, in which case, the fetch-execute cycle simply might not apply, although it still makes sense to use the fetch-execute metaphor to explain internal operations of those devices to novices.

All the above exceptions notwithstanding, computer scientists are content to refer to the fetch-execute cycle. If more clarity is preferred for a novice, decode might be a good metaphor to mention separately.

The Dark Side

Somewhere along the line, computer literacy instruction added an unfortunate twist to the machine cycle. It has become common to refer to the steps that comprise the machine cycle as fetch, decode, execute and store, e.g., [4, 7, 8, 9]. Of those, Smart Computing [8] complicates the issue further by stating the last two steps as "executing the code and then storing that code [SIC]." That is either a typographical error on that web site or a gross misstatement. The "code" (i.e., computer instruction) would not be stored again after execution, as it continues to be stored in main memory and in the instruction register.

Only Shelly, et. al., [7] attempt to clarify that "*storing*, in this context, means writing the result to memory (not to a storage unit)." Despite that attempt to clarify, confusion remains. If that reference is to a command that requires a calculated result to be retained, it would be retained in a register (usually known as the "accumulator"). It takes an additional machine language command to move the contents of the accumulator to memory. If the reference is to RISC architecture, the writing would not likely be to main memory but to a buffer.

Obviously, the problem with adding the term "store" into the metaphor is that it misleads and inhibits rather than enhances understanding. In most of the references cited in this paper, "store" refers to writing data to secondary storage devices such as disk. Such writing occurs when the instruction being handled is indeed a "store" and that occurs just as a fraction of instructions in a typical program. Store is only one form of execute, as are add, read, and all the other instructions in the machine's language.

If by "store" the writers mean "retaining an outcome of an execution," then they confuse the reader by using the term "store." Yet, even if that is their meaning, there are some instructions that produce no result to be retained. The extreme is the "no op" instruction which results only in a lapse of time. Moreover, retaining the outcome, like decoding, is not truly an identifiably separate step, depending on the architecture.

In summary, whereas adding "decode" to the "fetch-execute" metaphor might enhance it; it is not truthful. Worse, adding "store" diminishes the metaphor and potentially inhibits comprehension. Moreover, adding additional steps to the machine cycle doesn't help the novice at all to gain a better understanding. Rather, it builds confusion.

CONCLUSION

This paper has presented a number of metaphors commonly used in the information technology field. Some are good, others are not. Some enhance understanding, others inhibit it. Table 1 is offered as a clarification of this author's arbitrary use of those descriptors.

Table 1. Metaphor Classification

Category	Characteristic	Example
Good	Enhances understanding	Recycle bin/trash can
Incomplete	Promote understanding	Files-folders
Weak	Benign	Mouse, Desktop
Premature	Ineffective	Information Superhighway
Misunderstood	Confusing	World Wide Web
Bad, harmful	Inhibits understanding	Fetch-decode-execute
Out of control	Diminish understanding	Fetch-decode-execute-store

In particular, the fetch-execute cycle was examined at length. The metaphors of "fetch" and "execute" are much easier to understand for most novices than are the underlying electronic concepts. Adding "decode" to the mix, while not really a separate step, arguably might enhance the understandability of the overall metaphor.

Conversely, including any more steps in the fetch-execute cycle does not promote understanding. Rather, it confuses the issue. The more additional steps that are added to the metaphor, the further from universal reality the metaphor becomes. As a case in point, adding a fourth step of "store" is only correct a portion of the time and only for a subset of all computer systems. In short, the metaphor becomes harmful as it inhibits understanding. As such, it is an illustration of a metaphor gone wild.

REFERENCES

1. Brookshear, G. (2007). *Computer Science: An Overview, 9th ed*. Addison-Wesley Publishers.
2. Carpenter, D. A. (1995, March 2). "Information superhighway still bumpy." *Kearney Hub, 4A*.
3. Clinton, W. J., & Gore, A. Jr. (1997, Jul). A framework for global electronic commerce. Retrieved April 2005 from www.iitf.nist.gov/eleccomm/ecomm.
4. Computer User (2004). High Tech Dictionary: Machine Cycle. Retrieved March 13, 2006 from http://www.computeruser.com/resources/dictionary/definition.html?lookup=2948
5. Englander, I. (2000). *The Architecture of Computer Hardware and Systems Software, 2nd ed.* New York: Wiley.
6. FOLDOC: Free On-line Dictionary of Computing (1988, June 25). Fetch-execute cycle. Retrieved on March 13, 2006 from http://foldoc.org/?fetch-execute+cycle.
7. Shelly, G. B., Cashman, T. J., Vermaat, M. E. (2007). *Discovering Computers 2007: Gateway to Information,*.188. Boston: Thompson Course Technology.
8. Smart Computing. (2006). Machine cycle. Retrieved March 13, 2006 from http://www.computerhope.com/jargon/m/machcycl.htm.
9. Smith, J. (2005, Apr 8). Computer basics 4– processing: Machine cycle. *Jan's Computer Literacy* Retrieved Mar 13, 2006 from http://www.jegsworks.com /Lessons/lesson4/lesson4-4.htm.
10. Turing, A.M. (1949). Computing machinery and intelligence. In Laplante, P., Ed. (1996). *Great Papers in Computer Science,* 628-646. Minneapolis: West Publishing Company.
11. Von Neumann, J. (1945). First draft of a report on the EDVAC. In Laplante, P., Ed. (1996). *Great Papers in Computer Science,* 208-218. Minneapolis: West Publishing Co.
12. Webopedia.com. (2003, Mar 18). Instruction cycle. Retrieved March 13, 2006 from

http://www.webopedia.com /TERM/I/instruction_cycle.html.
13. Wikopedia, The Free Encyclopedia. (2006 Mar 3). The fetch-execute cycle. Retrieved March 13, 2006 from http://en.wikipedia.org/wiki/Fetch-execute_cycle.
14. Wilkes, M. V. (1951). The best way to design an automatic calculating machine. In Laplante, P., Ed. (1996). *Great Papers in Computer Science* (pp. 277-285). Minneapolis: West Publishing Company.
15. Williams, M. R. (1985). *A History of Computing Technology,* 122, 302. Englewood Cliffs, NJ: Prentice-Hall.